Práctica 1. Descomposición de Gauss.

Implementar el método de eliminación gaussiana para poder resolver sistemas del tipo $Ax = b$.

a) Para una matriz cuadrada A de tamaño $n \times n$, deseamos obtener su descomposición $PA = LU$, con P matriz de permutaciones, L matriz triangular inferior y U matriz triangular superior. Lo implementaremos en el **APÉNDICE A**.

La función nos debe devolver, dada A:

- p el vector puntero con n componentes que contenga las permutaciones de las filas que definen P
- Una nueva A, matriz cuadrada $n \times n$ que, al aplicarle la permutación p, contenga la matriz L (sin los unos de la diagonal) y la matriz U, sobreescribiendo sobre la A original para que el código sea más eficiente.

b) A continuación, utiliza el programa para resolver el siguiente sistema de ecuaciones:

$$y + 2z + t = 1$$
$$x + 2y + z + 3t = 1$$
$$x + y - z + t = 1$$
$$y + 8z + 12t = 1$$

```
%Práctica 1. Jaime Rodríguez.

A = [0, 1, 2, 1; 1, 2, 1, 3; 1, 1, -1, 1; 0, 1, 8, 12]
```

```
A = 4×4
     0     1     2     1
     1     2     1     3
     1     1    -1     1
     0     1     8    12
```

```
b = [1, 1, 1, 1]
```

```
b = 1×4
     1     1     1     1
```

```

tic
%LU equivale a la matriz L+U-I, que sustituye en memoria a A.
%p es el vector de permutación.
[LU,p] = factoPALU(A)
```

```
LU = 4×4
     0     1     2     1
     1     2     1     3
     1    -1     0    -1
```

```
     0     1     6    11
p = 1×4
     2     1     4     3
```

```
n = size(b,2);

% Ax=b ~ PAx=Pb ~ LUx=Pb ~ Ly=Pb
% Resolvemos Ly=Pb empezando por y(1) y acabando en y(n) mediante
% retrosubstitución.
y = 1:n;
y(1) = b(p(1));
for i = 2:n
    y(i) = b(p(i)) - sum(LU(p(i), 1:i-1) .* y(1:i-1));
end

%Ahora resolvemos Ux=y empezando por x(n) y acabando en x(1) también
%mediante retrosubstitución.
x = 1:n;
x(n) = y(n) / LU(p(n),n);
for i = n-1:-1:1 %[n-1,n-2,...,2,1]
    x(i) = (y(i) - sum(LU(p(i), i + 1: n) .* x(i+1:n))) / LU(p(i), i);
end
tPALU = toc;
disp("Solución usando la factorización PALU: ");
```

```
Solución usando la factorización PALU:
```

```
x = x'
```

```
x = 4×1
    5.5000
   -1.6667
    1.8333
   -1.0000
```

c) Compara el resultado con el obtenido por el comando \ de MATLAB.

```
%A\b calcula A^{-1} * b
%Ax=b ~ (A^-1)Ax=(A^-1)b ~ x=(A^-1)b
A = [0,1,2,1;1,2,1,3;1,1,-1,1;0,1,8,12];
b = [1;1;1;1];
disp("Solución usando el comando \ de división de matriz por la izq. de
MATLAB: ");
```

```
Solución usando el comando \ de división de matriz por la izq. de MATLAB:
```

```
tic
A\b
```

```
ans = 4×1
    5.5000
   -1.6667
```

```
    1.8333
   -1.0000
```

```
tDivIzq = toc;
```

d) Por último, compara los tiempos de ejecución de resolver el sistema haciendo uso del código del apartado $b)$ y de resolverlo mediante \.

```
disp("Tiempo al resolverlo mediante factorización PALU: " + tPALU);
```

```
Tiempo al resolverlo mediante factorización PALU: 0.013519
```

```
disp("Tiempo al resolverlo usando \: " + tDivIzq);
```

```
Tiempo al resolverlo usando \: 0.003136
```

Apéndice A. Factorización $PA = LU$

```
function [A,p]=factoPALU(A)
n = size(A,1);

%inicializamos el vector de permutación
p = (1:n);

for k = 1:n
    %Para cada columna, permutamos el elemento de máximo módulo con el
    %elemento de la diagonal para usarlo como pivote (De esta forma
    %evitamos errores al tratar con inversos de elementos pequeños).
    [a,l] = max(abs(A(p(k:n),k)));
    l = l + k - 1;
    if a ~= 0
        t = p(l);
        p(l) = p(k);
        p(k) = t;

        % f_i = f_i - ( A_{ik} / pivote ) * f_k
        for i = k + 1:n
            A(p(i),k) = A(p(i),k) / A(p(k),k);
            for j = k + 1:n
                A(p(i),j) = A(p(i),j) - A(p(i),k) * A(p(k),j);
            end
        end
    end
end

end
```